Server-Assisted Generation of a Strong Secret from a Password

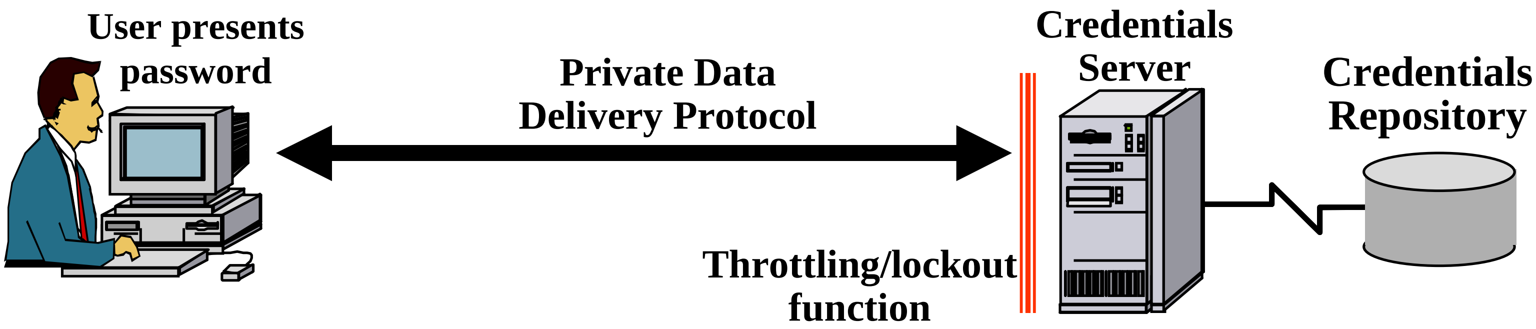
Warwick Ford, VeriSign, Inc.

(Joint research with Burt Kaliski, RSA Laboratories)

Requirement

- User who roams between client terminals needs to
  - obtain private key or data
  - strongly authenticate to application servers
- No local stored state
- No smartcards
- Private data downloaded from online *credentials server*

Traditional Credentials Server Solution

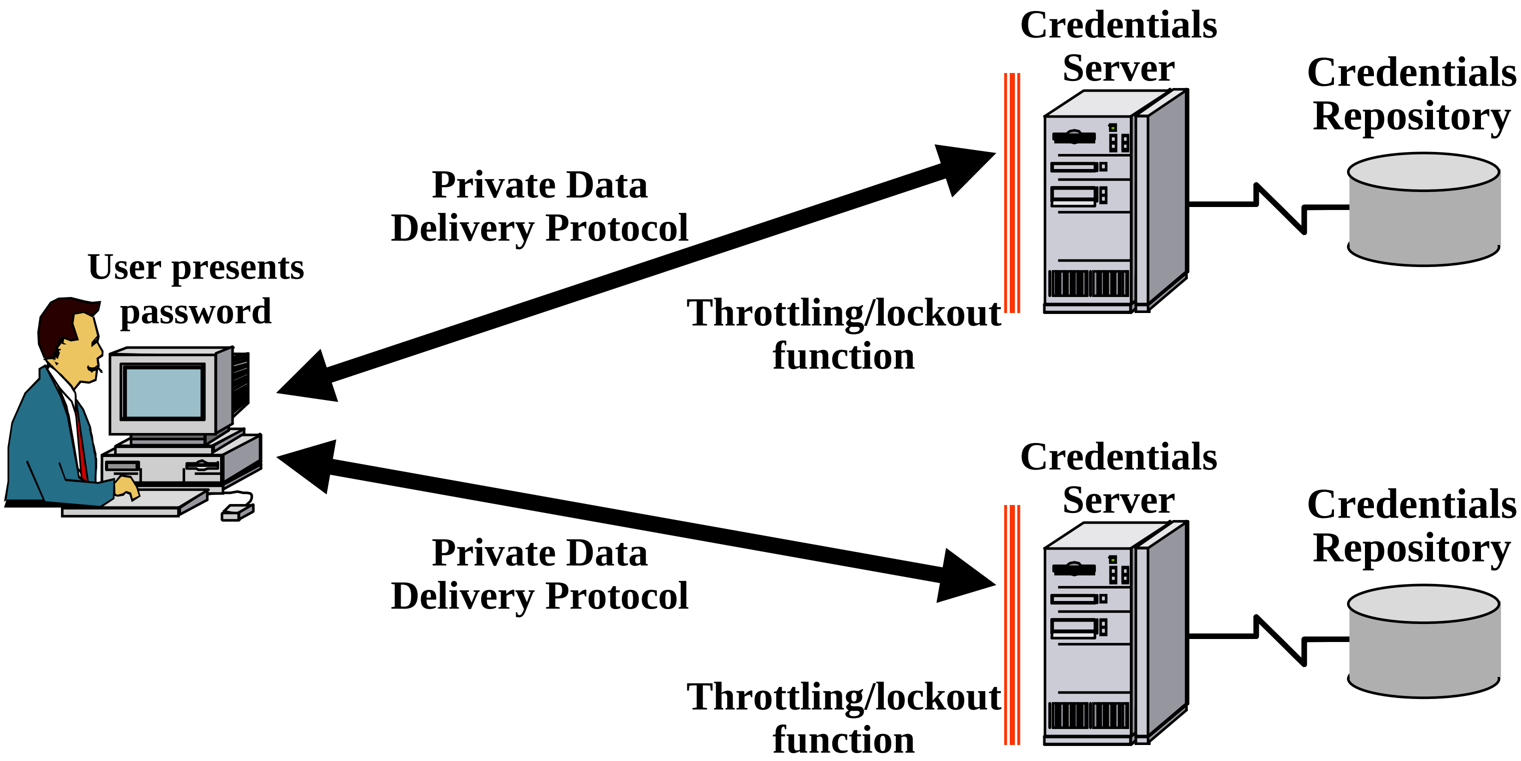


- Surveyed in Perlman & Kaufman, NDSS '99
  - Examples EKE, SPEKE
- Protocol exposes no information about private data
- Throttling/lockout:
  - Limits password guessing
  - Makes friendly passwords possible
  - Based on failed password authentications

Weakness in Traditional Design

- If server compromised, attacker can potentially:
  - Attack credentials database, e.g., password verifiers by exhaustive attack (even if passwords not determinable directly)
  - Disable throttling/lockout and exhaustively attack with password guesses
- Vulnerable to password attack
- Password exposure means private data exposure
- Many users may be compromised in one attack

Solution - Multiple Servers


Credentials Server
Credentials Repository
Private Data Delivery Protocol
User presents password
Throttling/lockout function
Credentials Server
Credentials Repository
Private Data Delivery Protocol
Throttling/lockout function

- Objective: Compromise of one server exposes neither private data nor password
- Not as easy as it looks
  - Ordinary secret-sharing not adequate if servers have to verify passwords

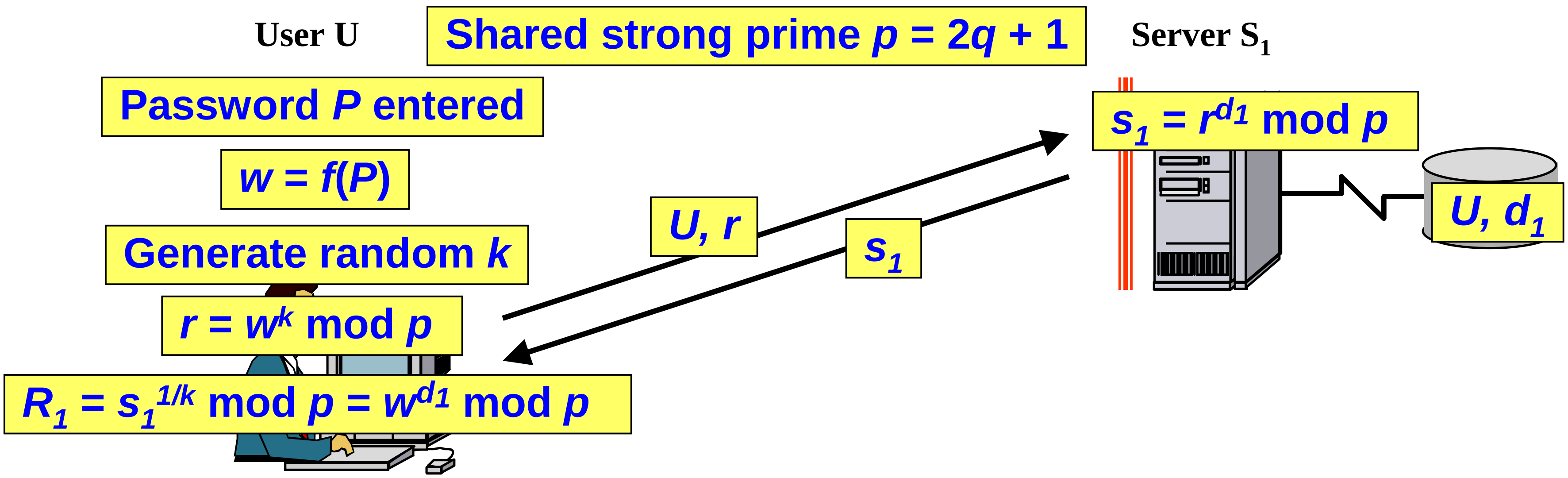
Basic Approach

- Client generates strong master secret *K* via interaction with two or more servers
- Client proves successful regeneration of *K* to all servers
- *K* can unlock encrypted private data or facilitate authentication to other servers
- No server can learn *K* or password

In More Detail…

- Pre-knowledge
  - User knows password P
  - Each server S_i holds its own secret d_i for that user
  - Each S_i also holds its own strong verifier K_i for K
- Client generates strong master secret K
  - For each S_i, client computes strong secret R_i
    - via a password hardening transaction depending on P and d_i
    - subject to throttling/lockout
  - Combines all the R_i to give K
- Client proves successful regeneration of K to servers
  - For each server S_i generates strong verifier K_i from K
  - Demonstrates knowledge of K_i to server S_i
- K can unlock encrypted private data or facilitate authentication to other servers

Secret-Strengthening Protocol



User U

Shared strong prime $p = 2q + 1$

Server S_1

Password P entered

$w = f(P)$

Generate random k

$r = w^k \bmod p$

$R_1 = s_1^{1/k} \bmod p = w^{d_1} \bmod p$

U, r

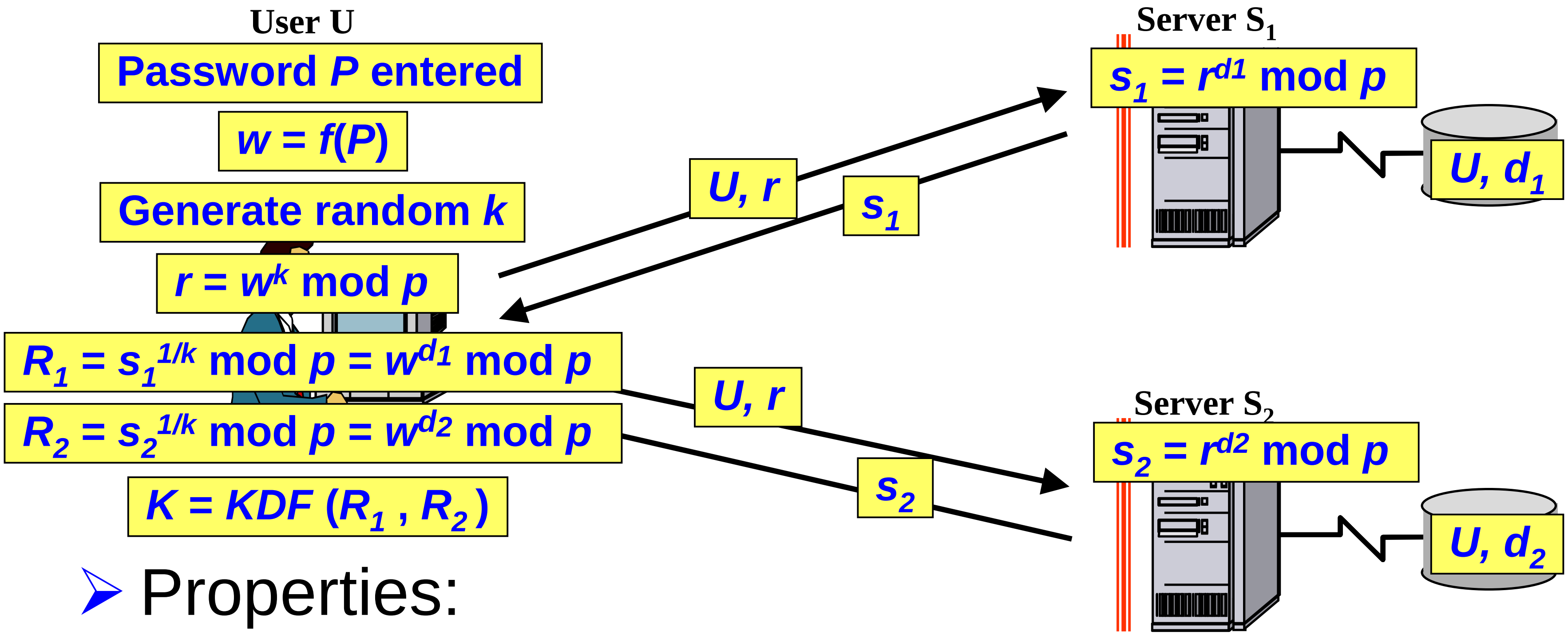
s_1

$s_1 = r^{d_1} \bmod p$

U, d_1

- Properties:
  - R_1 is a strong secret
  - Observer cannot feasibly learn R_1 ,d_1 or P
  - Server cannot feasibly learn R_1 [or P ?]
  - Same R_1 always generated for same P

Do It with Two Servers



User U

Password P entered

$w = f(P)$

Generate random k

$r = w^k \bmod p$

$R_1 = s_1^{1/k} \bmod p = w^{d1} \bmod p$

$R_2 = s_2^{1/k} \bmod p = w^{d2} \bmod p$

$K = KDF\,(R_1\,,R_2\,)$

U, r

s_1

Server S_1

$s_1 = r^{d1} \bmod p$

U, d_1

U, r

s_2

Server S_2

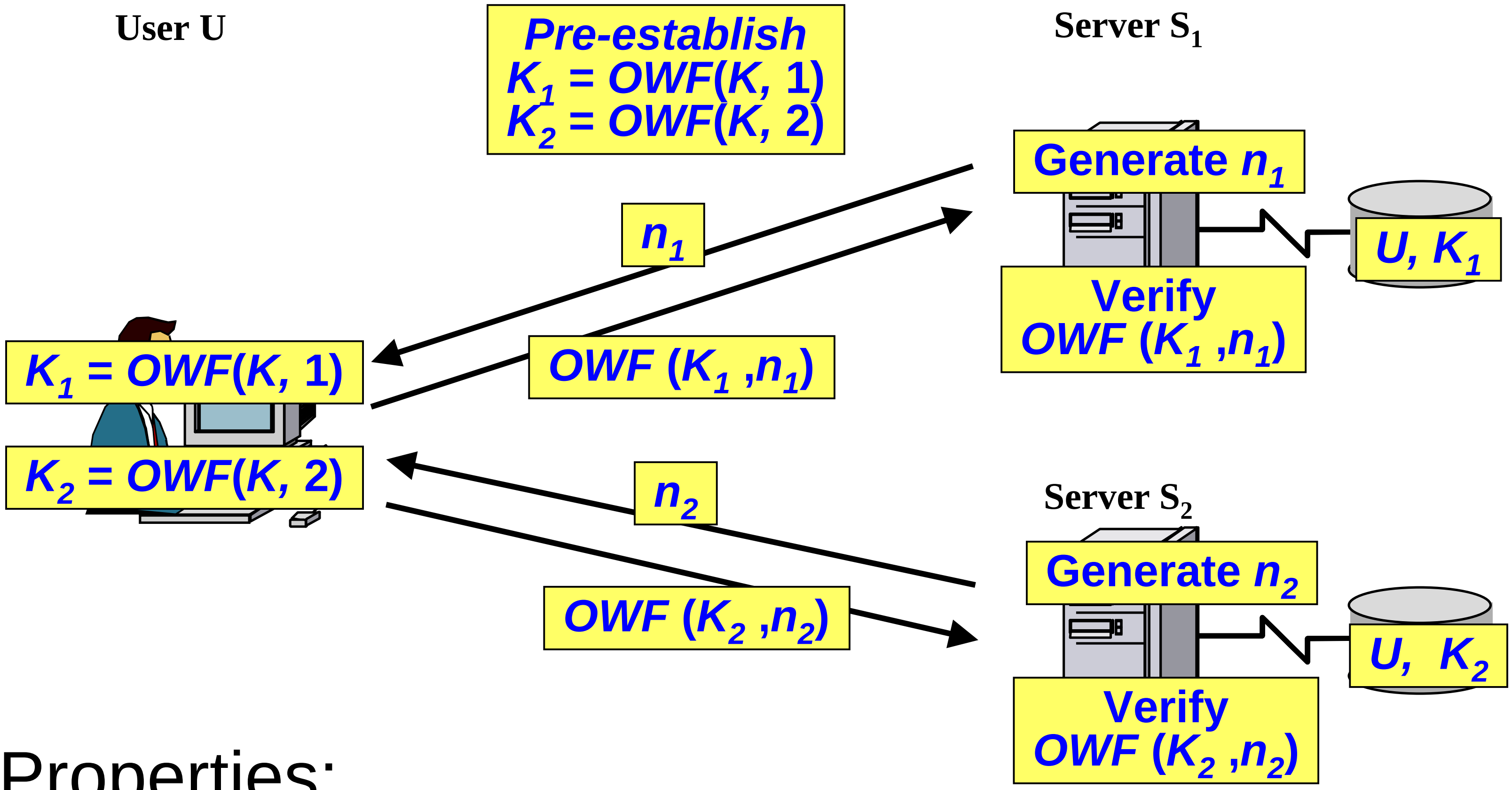
$s_2 = r^{d2} \bmod p$

U, d_2

- Properties:
  - *K* is a strong secret
  - Observer cannot feasibly learn *K* or *P*
  - Neither server can feasibly learn *K* or *P*
  - Same *K* always generated for same *P*
  - Both servers need to cooperate for *K* to be generated

VeriSign

Now Prove It was Successful



User U

Pre-establish
K_1 = OWF(K, 1)
K_2 = OWF(K, 2)

Server S_1

Generate n_1

n_1

U, K_1

Verify
OWF (K_1 ,n_1)

K_1 = OWF(K, 1)

OWF (K_1 ,n_1)

K_2 = OWF(K, 2)

n_2

Server S_2

Generate n_2

U, K_2

OWF (K_2 ,n_2)

Verify
OWF (K_2 ,n_2)

- Properties:
  - Each server gets proof that client knows *K*
  - Server's knowledge of K_i does not feasibly assist determining *K* (or password)

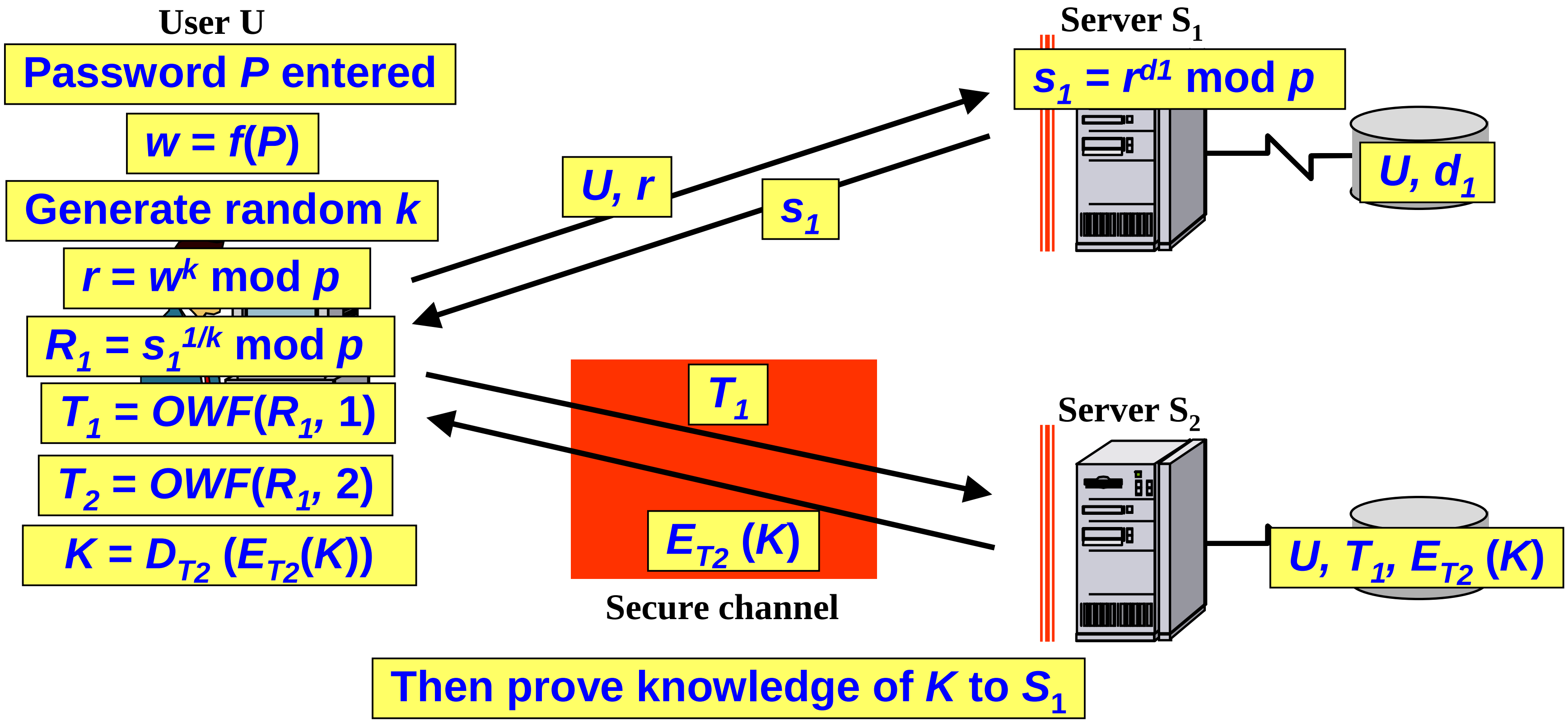
Some Variants

- Other secret-strengthening protocols
  - ECC variant is obvious
  - RSA-based also exists
- Other verification methods
  - *K* decrypts a private digital signature key; signed nonce proves regeneration to server holding public key
- Use threshold functions in combining hardened passwords
- Use other functions of master secret to authenticate to other (application) servers

A Special Case Variant

- Client interacts with password hardening server S_1 to obtain R_1
- Client uses T_1 derived from R_1 to authenticate to a second server S_2
- S_2 confidentially delivers to client: secret K encrypted under T_2 derived from R_1
- Client decrypts K
- Client verifies to S_1 by proving regeneration of K

Special Case Variant - Protocol



User U

Password *P* entered

$w = f(P)$

Generate random *k*

$r = w^k \bmod p$

$R_1 = s_1^{1/k} \bmod p$

$T_1 = OWF(R_1, 1)$

$T_2 = OWF(R_1, 2)$

$K = D_{T_2}(E_{T_2}(K))$

U, r

s_1

Server S_1

$s_1 = r^{d1} \bmod p$

U, d_1

T_1

$E_{T2}(K)$

Secure channel

Server S_2

$U, T_1, E_{T2}(K)$

Then prove knowledge of *K* to S_1

- Properties:
  - Attractive when S_2 already exists (e.g., SSL or SPEKE server)
  - Adding one password hardening server S_1 provides the requisite added strength

The Fundamental Characteristics

- Must recover a master secret using more than one independent server
  - all of which contribute to recovering the secret
  - all of which employ throttling/lockout
- At least one secret-contributing server must use secret-strengthening
- Must prove successful regeneration of a strong secret to at least two verification servers

Non-Repudiation Ramifications

- Single server design is weak wrt non-repudiation
  - user can plausibly claim that insider/penetrator at the server recovered the private key and signed
- The multi-server design significantly improves non-repudiation
  - it is much harder to mount a plausible argument that independently controlled servers colluded
- But, claims of non-repudiability still rest on confidence that the client terminal is secure
  - there is no silver bullet for this concern

Summary of the Technology

- Traditional credentials server architecture is vulnerable to server compromise and exhaustive password guessing against stored password-derived values
  - Server vulnerability raises security concerns and kills non-repudiation
- Need multiple independent servers contributing to secret regeneration
  - Each must independently throttle/lockout
- Need password hardening as a basis of establishing strong secret from weak secret

VeriSign

Deployment Status

- Current-shipping VeriSign enterprise PKI offering includes the option:
  - Two-server secret-strengthening technology to support protection of private key plus arbitrary user data
  - Servers may be operated by Enterprise and/or VeriSign
- Alternative packagings (e.g., for SSO, Aggregation) in development

For More Information

- See Ford/Kaliski WETICE 2000 paper at:

  http://www.verisign.com/repository/pubs/roaming.pdf

- Contact details:

Warwick Ford, VeriSign, Inc.
E-mail: wford@verisign.com
Tel: (781) 245 6996 x225